A Comparison of Greedy Search Algorithms

Christopher Wilt, Jordan Thayer and Wheeler Ruml

University of New Hampshire

The Problem

There are many algorithms for solving shortest path problems.

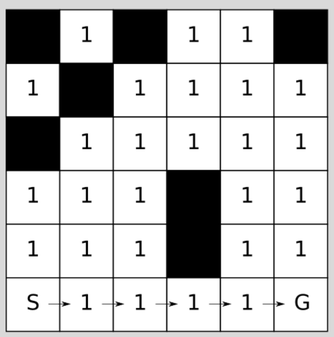
When given a problem, which algorithm should we use?

Approach

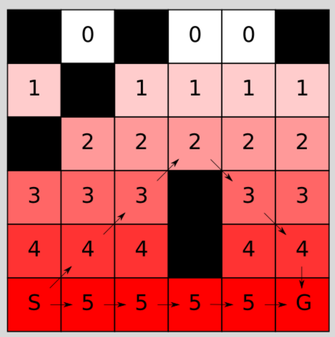
We considered the following domains:

Domain	Cycles	Dead Ends	Unit Cost	Unique States
TSP	None	No	No	1×10^{30}
Grid Unit	Short	Yes	Yes	1×10^{6}
Vacuum	Short	Yes	Yes	6×10^{23}
Grid Life	Short	Yes	No	1×10^{6}
Pancake	Short	No	No	3×10^{6}
Robot	Long	Yes	No	2×10^{11}
15 Puzzle	Long	No	Yes	6×10^{11}
48 Puzzle	Long	No	Yes	3×10^{62}

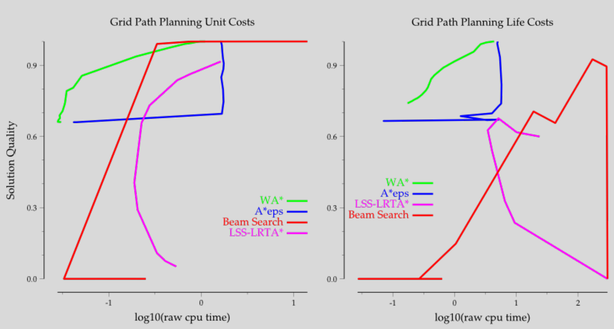
Unit and Non-Unit Cost



Best path: 5
Shortest path: 5



Shortest path: 25
Best path: 21



Best-first searches handle non-unit costs well.

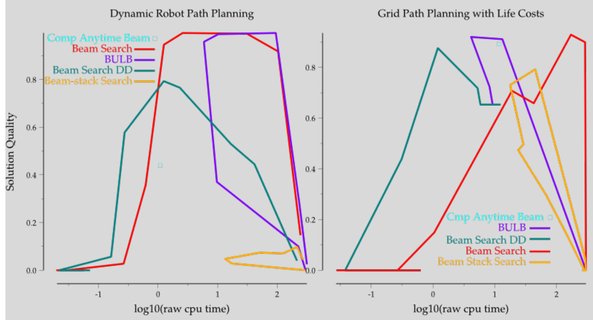
Duplicate Nodes and Cycles

In domains with cycles, beam searches need to detect duplicates to break cycles.

Beam Search on a sliding tile puzzle instance

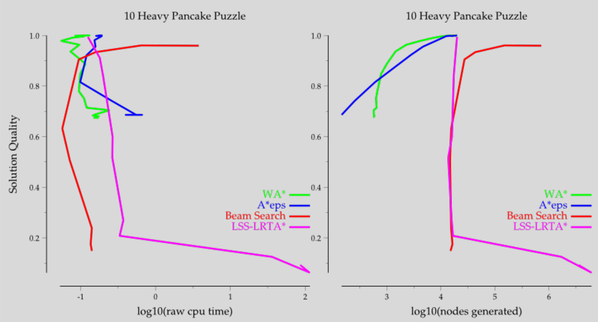
Algorithm	Sol. Len	Nodes	Dups	Unique
closed list	511	5312	65	5247
no closed(20000)	n/a	20000	17533	2467
no closed(40000)	n/a	40000	37533	2467

Dropping Duplicates

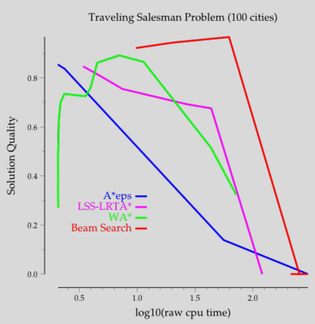


Dropping duplicates can cause odd scaling behavior in beam search.

Node Generation Time



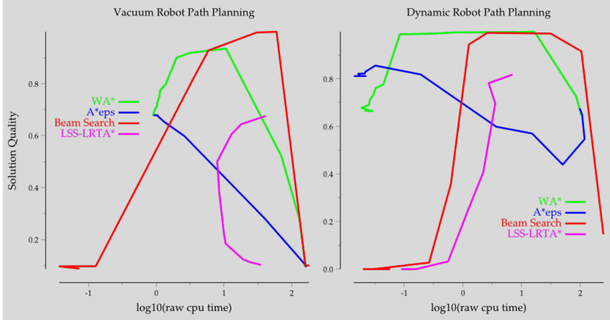
Beam search is dominated on time, not nodes.



Pancake nodes are small and have a cheap heuristic, so generating nodes isn't a problem.

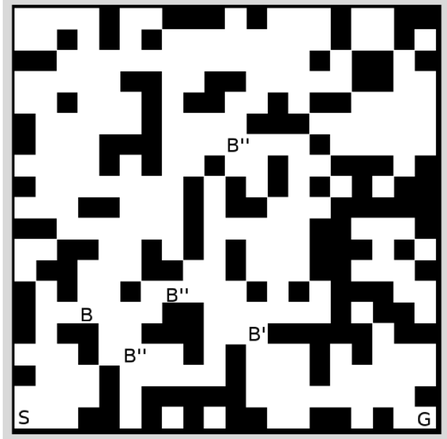
TSP nodes take longer to generate, so beam searches find their first solution much later.

Dead Ends



Beam searches with small beams fail if there are dead ends

Bottleneck States (Landmarks)

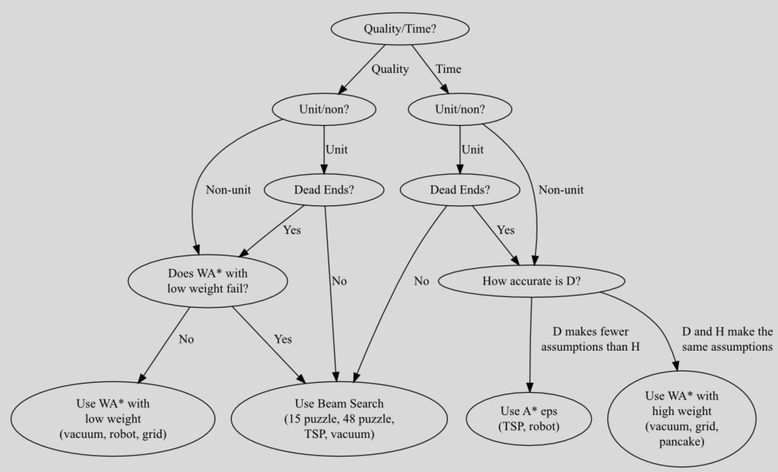


Sometimes important nodes like the ones with a B are pruned.

Pruning B, B', or all three B'' nodes will make it impossible to find a solution.

Rules of Thumb

A decision tree to help us choose an algorihtm



This tree fits the available data, but it needs more data to be more robust.